

Хождение по граблям PDO

Валерий Горбачев

О том, что скрывают за собой современные PHP ORM.
Пособие для любопытствующих





Кто я?

Меня зовут Валерий Горбачев

Работаю программистом в Delivery Club

Использую: PHP, JS, SQL



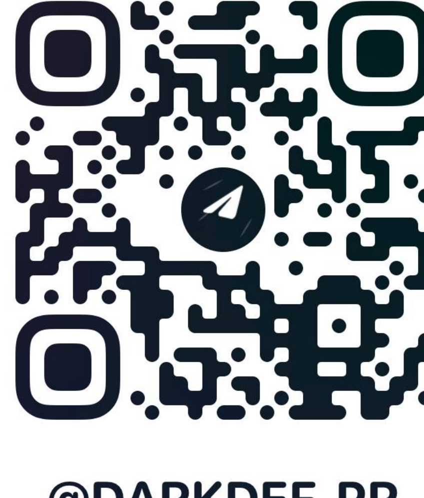


- немного организатор (митапы по PHP в Краснодаре)
- немного спикер (локальные митапы)
- участник команды разработки Yii3 framework
- основатель чата @phpkrd

План доклада

- Исторический экскурс
- Грабли
- Костыли

  …
- Вывод



Disclaimer:

Все ошибки случайны и связаны с плохой памятью докладчика, а не совершены по злому умыслу.

Как я изучал БД

Классическая кривая обучения:

- Отрицание
- Гнев
- Торг
- Депрессия
- Принятие



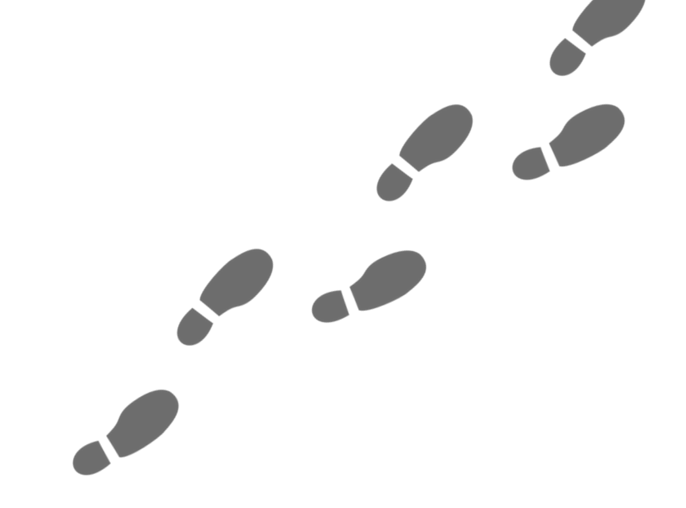
Как я изучал БД

~~Классическая кривая обучения:~~

- ~~Отрицание~~
- ~~Гнев~~
- ~~Торг~~
- ~~Депрессия~~
- ~~Принятие~~

Как было на самом деле:

- Данные в файлах
- Файловые БД (dbf, dBase IV)
- Изучение SQL (книги, документация)



Запросы к БД — это просто

Я уже умею писать SQL-запросы:

```
$sql = "SELECT * FROM products WHERE id = {$id}"
```

и даже иногда забочусь о безопасности

```
$sql = 'SELECT * FROM products WHERE id = ' . @intval($id);
```

Не «само» развитие

Когда мне приходилось сохранять строковые данные, я делал вот так:

```
$sql = "SELECT * FROM products WHERE name = '{$name}'";
```

однажды меня взломали, и **я узнал про SQL-уязвимости**

Не «само» развитие

Когда мне приходилось сохранять строковые данные, я делал вот так:

```
$sql = "SELECT * FROM products WHERE name = '{$name}'";
```

однажды меня взломали, и **я узнал про SQL-уязвимости**

Я начал делать вот так и делал так не один год:

```
$name = mysql_real_escape_string($name);
$sql = "SELECT * FROM products WHERE name = '". $name. "'";
```

Потом узнал, что MySQLi лучше (когда mysql_* задепрекейтили)

```
$stmt = $mysqli->prepare("SELECT * FROM products WHERE name = ?");
$stmt->bind_param("s", $name);
```

Про deprecated mysql_

- PHP 5.0 – появился ext/mysqli (2003)

Про deprecated mysql_

- PHP 5.0 – появился ext/mysqli (2003)
- PHP 5.1 – появление модуля PDO_MySQL (2005)

Про deprecated mysql_

- PHP 5.0 – появился ext/mysqli (2003)
- PHP 5.1 – появление модуля PDO_MySQL (2005)
- PHP 5.3 – появление mysqlnd (2009)

Про deprecated mysql_

- PHP 5.0 – появился ext/mysqli (2003)
- PHP 5.1 – появление модуля PDO_MySQL (2005)
- PHP 5.3 – появление mysqlnd (2009)
- PHP 5.* – переход с libmysqlclient (C++, Oracle) на mysqlnd (2013)

Про deprecated mysql_

- PHP 5.0 – появился ext/mysqli (2003)
- PHP 5.1 – появление модуля PDO_MySQL (2005)
- PHP 5.3 – появление mysqlnd (2009)
- PHP 5.* – переход с libmysqlclient (C++, Oracle) на mysqlnd (2013)
- PHP 5.5 – модуль mysql_* помечен устаревшим и удалён в версии 7 (2013)

… в это время я задумался о переходе на mySQLi/PDO (для старого кода)

	MySQLi	PDO_MySQL
Рекомендовано для новых проектов	Да	Да
API поддерживает асинхронные, неблокирующие запросы mysqlnd	Да	Нет
Постоянные (persistent) соединения	Да	Да
API поддерживает подготовленные запросы на стороне сервера	Да	Да
API поддерживает подготовленные запросы на стороне клиента	Нет	Да
Поддерживает всю функциональность MySQL 5.1+	Да	Большинство
API поддерживает множественные запросы	Да	Большинство

Используя PDO, можно легко сменить используемую СУБД © не я

к сожалению, я это запомнил...

И на этом можно было бы остановиться...

На новых проектах я использовал ORM из состава фреймворков

```
$userQuery = (new Query())->select('id')->from('user');
```

И на этом можно было бы остановиться...

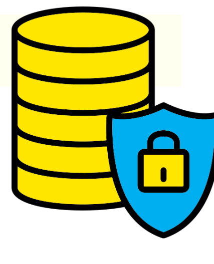
На новых проектах я использовал ORM из состава фреймворков

```
$userQuery = (new Query())->select('id')->from('user');
```

На старых проектах я перешёл на MySQLi / PDO

MySQLi — ведь это было максимально быстро и просто, но в биндинге можно легко запутаться.

```
$stmt->bind_param("ssssiisississ", $name, $title, ...);
```



ORM – лекарство от всех проблем

Основные преимущества ORM

- Код может быть абстрактным
- Нет нужды писать «сырые» запросы
- Порог вхождения гораздо ниже
- Код не слишком привязан к движку БД

Eloquent, Doctrine, Cycle, Propel, RedBeanPHP, YiiDB и т.д.

Недостатки ORM

Без обобщений — дальше я буду говорить, опираясь на опыт, полученный от использования ORM от Yii Framework 1/2

Ключевые недостатки

- Нет той же гибкости, как при написании «сырых» запросов
- Сложно писать сложные запросы
- Работает быстро, но немного медленнее нативных библиотек
- Иногда legacy по принципу «так сложилось»
- MySQL-first-подход

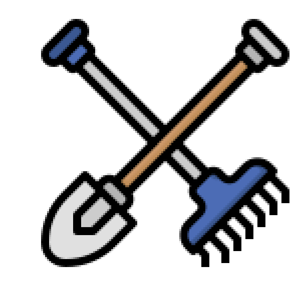
Хождение по граблям как стиль жизни

давайте погрустим вместе – мне мешают использовать ORM всю жизнь :(

- Clipper или приключения в сетях IPX/SPX
- C++ Builder и Paradox в мире блокировок
- Кто одержит победу: mysql vs mysqli
- T-SQL. Бизнес-логика в хранимых процедурах
- Oracle + OCI8. Взаимодействие через хранимые процедуры
- Инкапсуляция запросов в API с использованием MS SQL

…

- Гарри Поттер и Орден Феникса



Перемотка

Пропустим затянутый очерк о моей жизни и перейдём к сути:

- Познакомился с Александром Макаровым
- Поковырялся в Yii2 по части MS SQL
- Догадался, где пригодится поверхностный опыт использования разных БД
- Познакомился с Wilmer Arambula, и мы вместе рефакторим yiisoft/db
- В основе пакетов yiisoft/db лежит PDO…

Скучный пересказ документации

в которой можно найти ответы на большинство вопросов

PDO как глоток свежего воздуха

Подстановка данных может быть удобной…

Здесь могла бы быть картинка

с девушкой ведущей вас за руку

Но презентация про PDO,

поэтому без весёлых картинок

Использование PDO

В использовании PDO нет магии, и выглядит всё очень просто

```
/* Выполнение запроса с передачей ему массива параметров */
$sql = 'SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour';
$sth = $dbh->prepare($sql, ['calories' => 150, 'colour' => 'red']);
$sth->execute();
$red = $sth->fetchAll();
```

Пример взят из официальной документации

Огромные возможности PDO

PDO_CUBRID	Cubrid
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC и win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 и SQLite 2
PDO_SQLSRV	Microsoft SQL Server / SQL Azure

Другой «движок» базы данных

~~MySQL~~ PostgreSQL – отраслевой стандарт, который должен знать каждый уважающий себя web-разработчик.

Но это ведь не проблема, когда у тебя PDO?

Другой «движок» базы данных

~~MySQL~~ PostgreSQL – отраслевой стандарт, который должен знать каждый уважающий себя web-разработчик.

Но это ведь не проблема, когда у тебя PDO?

Но:

1. Запрос запросу рознь (**LIMIT 10 OFFSET 10** vs **LIMIT 10, 10**)
2. Универсального кода не бывает
3. Вообще – бывает, но работает неожиданно…

Добро пожаловать в жестокий мир PDO



ТОП-5

неиспользуемых режимов PDO::fetch

Из документации...

```
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* Извлечение всех оставшихся строк результирующего набора */
print("Извлечение всех оставшихся строк результирующего набора:\n");
$result = $sth->fetchAll();
print_r($result);
?>
```

Самые используемые: **FETCH_ASSOC** или **FETCH_OBJ**

FETCH_KEY_PAIR

```
SELECT username, email FROM users
```

```
[
    'login1' => 'email1@mail.ru',
    'login2' => 'email2@mail.ru',
]
```

Просто key-value. Следит, чтобы в выборке было только два поля. Перезаписывает последним ключом.

FETCH_UNIQUE

Сочетание: **PDO::FETCH_UNIQUE | PDO::FETCH_ASSOC**

```
SELECT `id`, `username`, `email` FROM `users`
```

```
[
  1 => ['username' => 'login101', 'email' => 'email1@mail.ru', ],
  2 => ['username' => 'login102', 'email' => 'email2@mail.ru', ],
]
```

Всё как и в предыдущем случае, только значения в виде массива. Следим за уникальностью сами.

FETCH_GROUP

Сочетание: **PDO::FETCH_GROUP | PDO::FETCH_ASSOC**

```
SELECT `group`, `id`, `username` FROM `users`
```

```
[
    'reader' => [
        ['id' => 1, 'username' => 'login101'],
        ['id' => 3, 'username' => 'login103'],
    ],
    'writer' => [
        ['id' => 2, 'username' => 'login102'],
        ['id' => 4, 'username' => 'login104'],
    ],
]
```

А вот так — группировка по первому значению в выборке.

FETCH_NAMED вместо FETCH_ASSOC

```
SELECT u.`name`, u.`email`, g.`name`
FROM `users` u
INNER JOIN `groups` g ON g.`id` = u.`group_id`
```

```
$stmt->fetch(PDO::FETCH_NAMED);
/*[
    'name' => [
        0 => 'login101',
        1 => 'writer',
    ],
    'email' => 'email1@mail.ru',
]*/
```

А вот так мы не потеряем значение, если поле в выборке несколько раз.

FETCH_CLASSTYPE

```
SELECT
    CONCAT(UCASE(LEFT(`group`, 1)), SUBSTRING(`group`, 2)),
    `username`, `email`
FROM `users`
```

```
$stmt->fetch(PDO::FETCH_CLASS | PDO::FETCH_CLASSTYPE);
/*
object(Reader)#3 (1) {
  ["username"] => string(8) "login101",
  ["email"] => string(14) "email1@mail.ru",
}*/
```

Пахнуло ~~дымом с завода~~ фабрикой? А будет ли это работать в PHP9?

Интересно? Вот тут детально: https://phpdelusions.net/pdo/fetch_modes

А есть ли еще «занимательные флаги»

- **PDO::ATTR_ORACLE_NULLS** – "" => NULL, NULL => "" (не только Oracle)
- **PDO::ATTR_CASE** – регистр имён столбцов (MySQL 8.0.21)
- **MYSQL_ATTR_LOCAL_INFILE_DIRECTORY** (с PHP 8.1)

```
LOAD DATA LOCAL INFILE 'path/to/file/file.txt'
```

А есть ли еще «занимательные флаги»

- **PDO::ATTR_ORACLE_NULLS** – "" => NULL, NULL => "" (не только Oracle)
- **PDO::ATTR_CASE** – регистр имён столбцов (MySQL 8.0.21)
- **MYSQL_ATTR_LOCAL_INFILE_DIRECTORY** (с PHP 8.1)

```
LOAD DATA LOCAL INFILE 'path/to/file/file.txt'
```

Странные флаги:

- PDO::MYSQL_ATTR_MULTI_STATEMENTS
- PDO::ATTR_FETCH_CATALOG_NAMES
- PDO::ATTR_FETCH_TABLE_NAMES

Мне кажется, что PDO ждёт большое будущее… **breaking change** в PHP 9

Регистр и MySQL 8.0.21

Что можно ожидать в результате выполнения запроса?

```
SELECT `constraint_name` from `information_schema`.`key_column_usage`;
/*
MySQL < 8.0.21 fields name:
array(1) {
  'constraint_name' => "name_of_contraint"
}

MySQL 8.0.21 fields name:
array(1) {
  'CONSTRAINT_NAME' => "name_of_contraint"
}
*/
```

И вот тут на сцену может выйти: PDO::ATTR_CASE => PDO::CASE_LOWER

https://github.com/yiisoft/yii2/issues/18171

Хранение JSON в BLOB

```
$sql_insert = <<<SQL
    insert into `pdo_types`(`blob_col`) values(:blob_data);
SQL;

$sql_read = <<<SQL
    select `blob_col` from `pdo_types`;
SQL;
```

* сейчас будем делать примерно как в документации

** и да, мы не используем CLOB, JSONB и т.п. спец.типы для хранения

MySQL

Вставляем данные

```
$blobData = "test `s_t_r_i_n_g`";
$pdo->prepare($sql_insert)->execute([':blob_data' => $blobData]);
```

Читаем данные

```
$stmt = $pdo->query($sql_read);
$result = $stmt->fetch(PDO::FETCH_COLUMN);
var_dump($result);
```

Радуемся

```
string(18) "test `s_t_r_i_n_g`"
```

PostgreSQL

Вставляем данные

```
$blobData = "test `s_t_r_i_n_g`";
$pdo->prepare($sql_insert)->execute([':blob_data' => $blobData]);
```

Читаем данные

```
$stmt = $pdo->query($sql_read);
$result = $stmt->fetch(PDO::FETCH_COLUMN);
var_dump($result);
```

Удивляемся

```
resource(9) of type (stream)
```

PostgreSQL

Вставляем данные

```
$blobData = "test `s_t_r_i_n_g`";
$pdo->prepare($sql_insert)->execute([':blob_data' => $blobData]);
```

Читаем данные

```
$stmt = $pdo->query($sql_read);
$result = $stmt->fetch(PDO::FETCH_COLUMN);
var_dump($result);
```

Удивляемся

```
resource(9) of type (stream)
```

Читаем документацию и радуемся

```
var_dump(stream_get_contents($result));
```

Немного экзотики – Oracle

Вставляем

```
$blobData = "test `s_t_r_i_n_g`";
$pdo->prepare($sql_insert)->execute([':blob_data' => $blobData]);
```

Удивляемся

```
PHP Fatal error:  Uncaught PDOException: SQLSTATE[HY000]: General
error: 1465 OCIStmtExecute: ORA-01465: invalid hex number
```

Читаем документацию

> *В случае с базами Oracle требуется* ***несколько иной*** *синтаксис для извлечения содержимого файла и помещения в базу. Также необходимо выполнять вставку в рамках транзакции, иначе вставленный LOB будет зафиксирован в базе с нулевой длиной, так как если не обозначить границы транзакции, изменения будут фиксироваться после каждого выполненного запроса.*

Работающий код для вставки

```
$sql_insert = <<<SQL
    insert into "pdo_types"("blob_col") values(empty_blob()) returning
"blob_col" into :blob_data
SQL;

$blobData = "test `s_t_r_i_n_g`";
$fp = fopen('php://memory', 'rwb');
fwrite($fp, $blobData);
fseek($fp, 0);

$stmt = $pdo->prepare($sql_insert);
$pdo->beginTransaction();
    $stmt->bindValue(':blob_data', $fp, PDO::PARAM_LOB);
    $stmt->execute();
$pdo->commit();
```

Rewind?

```
$stmt = $pdo->query($sql_read);
$handle = $stmt->fetch(PDO::FETCH_COLUMN);

$contents = '';
while (!feof($handle)) {
    $contents .= fread($handle, 2);
}
rewind($handle);
while (!feof($handle)) {
    $contents .= fread($handle, 2);
}
```

```
test `s_t_r_i_n_g`test `s_t_r_i_n_g`
```

Работает и rewind и fseek

Чиним BLOB поля и Oracle

Если данные у вас не совсем бинарные, к примеру JSON/base64, то можно без транзакций и файлов.

```
$stmt = $db->prepare("INSERT INTO t1(blob_col)
        VALUES (TO_BLOB(UTL_RAW.CAST_TO_RAW(:blob_col)))");
$stmt->bindParam(':blob_col', $blobData);
```

* лучше, конечно, хранить base64 в CLOB, а json в JSON

Чиним BLOB поля и Oracle

Если данные у вас не совсем бинарные, к примеру JSON/base64, то можно без транзакций и файлов.

```
$stmt = $db->prepare("INSERT INTO t1(blob_col)
        VALUES (TO_BLOB(UTL_RAW.CAST_TO_RAW(:blob_col)))");
$stmt->bindParam(':blob_col', $blobData);
```

* лучше, конечно, хранить base64 в CLOB, а json в JSON

Изучаем особенности PDO дальше?



Получение последнего вставленного ID

Что может быть проще?

Пример взят из официальной документации

```
public PDO::lastInsertId(?string $name = null): string|false
```

Также как и это замечание



Замечание:
*В зависимости от драйвера PDO этот метод может вообще не выдать осмысленного результата, так как база данных **может не поддерживать** автоматического инкремента полей или последовательностей.*

MS SQL Server

> *I am running an insert query using PDO and then getting the newly created Id with lastInsertId(). This is all working on my localhost environment.*
> *When I move the exact same code onto a server, the lastInsertId() is always returning blank, even though the insert statement works and inserts the new row into the database.*

https://stackoverflow.com/questions/13747481/pdo-lastinsertid-not-working-for-ms-sql

MS SQL Server



I am running an insert query using PDO and then getting the newly created Id with lastInsertId(). This is all working on my localhost environment.
When I move the exact same code onto a server, the lastInsertId() is always returning blank, even though the insert statement works and inserts the new row into the database.

https://stackoverflow.com/questions/13747481/pdo-lastinsertid-not-working-for-ms-sql

Remarks

Поддержка PDO была добавлена в версии 2.0 Драйверы Microsoft SQL Server для PHP.
Между версиями 2.0 и 4.3 необязательным параметром является имя таблицы, а возвращаемым значением — идентификатор последней добавленной в указанную таблицу записи. Начиная с 5.0, как необязательный параметр рассматривается имя последовательности, а как возвращаемое значение — последовательность, которую добавили для указанного имени последовательности последней. Если имя таблицы указано для версий после 4.3, `lastInsertId` возвращает пустую строку. Последовательности поддерживаются только в SQL Server 2012 и более поздних версиях.

Смириться или приложить подорожник?

Починим **lastInsertId** для Microsoft SQL Server (старых версий)

```
class mssqlPDO extends \PDO
{
    /**
     * Returns value of the last inserted ID.
     * @param string|null $sequence the sequence name. Defaults to null.
     * @return int last inserted ID value.
     */
    public function lastInsertId($sequence = null)
    {
        $sql = 'SELECT CAST(COALESCE(SCOPE_IDENTITY(), @@IDENTITY) AS
bigint)';
        return $this->query($sql)->fetchColumn();
    }
}
```

Disclaimer

Решение по излечению lastInsertId может работать неожиданно…



For example, there are two tables, T1 and T2, and an INSERT trigger is defined on T1. When a row is inserted to T1, the trigger fires and inserts a row in T2. This scenario illustrates two scopes: the insert on T1, and the insert on T2 by the trigger.

https://docs.microsoft.com/ru-RU/sql/t-sql/functions/scope-identity-transact-sql?view=sql-server-2016

IDENT_CURRENT – returns the last identity value generated for a specific table in any session and any scope.

@@IDENTITY – returns the last identity value generated for any table in the current session, across all scopes.

SCOPE_IDENTITY – returns the last identity value generated for any table in the current session and the current scope.

Не используйте lastInsertId?

```
$pdo1->prepare($sql_insert)->execute([':n' => 101]);
$pdo2->prepare($sql_insert)->execute([':n' => 102]);

echo 'lastInsertId1 = '.var_export($pdo1->lastInsertId(), true).PHP_EOL;
echo 'lastInsertId2 = '.var_export($pdo2->lastInsertId(), true).PHP_EOL;
// lastInsertId1 = '1'
// lastInsertId2 = '2'
```

MySQL, PostrgeSQL, MS SQL – **PASSED**

Oracle – **FAIL**

```
PHP Fatal error:  Uncaught PDOException: SQLSTATE[IM001]: Driver does not
support this function: driver does not support lastInsertId()
```

Но если очень хочется...

Проверено с php8.0 + Oracle XE 11g r2 + instantclient21_3

```
$stmt1 = $pdo1->query('SELECT "pdo_types_SEQ".CURRVAL FROM DUAL');
$result1 = $stmt1->fetchColumn();

$stmt2 = $pdo2->query('SELECT "pdo_types_SEQ".CURRVAL FROM DUAL');
$result2 = $stmt2->fetchColumn();

// result1 = '1'
// result2 = '2'
```

Такой же костыль для MS SQL

Проверено с php8.0 + SQL Server 2017 (таблица без триггеров)

```
$stmt1 = $pdo1->query(
    'SELECT CAST(COALESCE(SCOPE_IDENTITY(), @@IDENTITY) AS bigint)');
$result1 = $stmt1->fetchColumn();

$stmt2 = $pdo2->query(
    'SELECT CAST(COALESCE(SCOPE_IDENTITY(), @@IDENTITY) AS bigint)');
$result2 = $stmt2->fetchColumn();

// result1 = '1'
// result2 = '2'
```

Но будет работать и со старыми версиями.

Непрошенный совет

И всё же не используйте lastInsertId, есть способы лучше:

- PostgreSQL – RETURNING "id"
- MS SQL – OUTPUT INSERTED.ID *
- Oracle – RETURNING id INTO :id
- MariaDB (с версии 10.5.0) – RETURNING `id`
- MySQL старых версий – никак

Только используя PDO с Oracle, я понял, зачем нужно bindParam

Еще немного непрошенных советов

- **PDO::MYSQL_ATTR_INIT_COMMAND** можно использовать для SET NAMES в сочетании с charset=utf8mb4 в DSN строке

Еще немного непрошенных советов

- **PDO::MYSQL_ATTR_INIT_COMMAND** можно использовать для SET NAMES в сочетании с charset=utf8mb4 в DSN строке
- Принудительно выключайте **PDO::ATTR_EMULATE_PREPARES**

Еще немного непрошенных советов

- **PDO::MYSQL_ATTR_INIT_COMMAND** можно использовать для SET NAMES в сочетании с charset=utf8mb4 в DSN строке
- Принудительно выключайте **PDO::ATTR_EMULATE_PREPARES**
- Именованые подстановки подставляются один раз (MySQL ↑)

Еще немного непрошенных советов

- **PDO::MYSQL_ATTR_INIT_COMMAND** можно использовать для SET NAMES в сочетании с charset=utf8mb4 в DSN строке
- Принудительно выключайте **PDO::ATTR_EMULATE_PREPARES**
- Именованые подстановки подставляются один раз (MySQL ↑)
- Не ленитесь: bindValue('attr', $value, **PDO::PARAM_INT**) лучше execute(['attr' => $value])

Еще немного непрошенных советов

- **PDO::MYSQL_ATTR_INIT_COMMAND** можно использовать для SET NAMES в сочетании с charset=utf8mb4 в DSN строке
- Принудительно выключайте **PDO::ATTR_EMULATE_PREPARES**
- Именованые подстановки подставляются один раз (MySQL ↑)
- Не ленитесь: bindValue('attr', $value, **PDO::PARAM_INT**) лучше execute(['attr' => $value])
- **MYSQL_ATTR_USE_BUFFERED_QUERY=false** для больших запросов (mysqlnd)

Еще немного непрошенных советов

- **PDO::MYSQL_ATTR_INIT_COMMAND** можно использовать для SET NAMES в сочетании с charset=utf8mb4 в DSN строке
- Принудительно выключайте **PDO::ATTR_EMULATE_PREPARES**
- Именованые подстановки подставляются один раз (MySQL ↑)
- Не ленитесь: bindValue('attr', $value, **PDO::PARAM_INT**) лучше execute(['attr' => $value])
- **MYSQL_ATTR_USE_BUFFERED_QUERY=false** для больших запросов (mysqlnd)
- Избегайте многострочных SQL-запросов в одном вызове. Но если уж так получилось:
  - execute() сообщит об ошибке только в первом запросе
  - Используйте nextRowSet() для выборки следующего набора
  - nextRowSet() – может вернуть пустой результат, который нельзя выбрать (columnCount())

Про nextRowSet()

Иногда не получается избежать многострочных запросов

```
$sql_insert = <<<SQL
SET NOCOUNT ON;
DECLARE @tmp TABLE ([id] int);
INSERT INTO [table]([val]) OUTPUT INSERTED.id INTO @tmp VALUES('12');
SELECT * FROM @temporary_inserted;
SQL;
```

```
$stmt = $pdo->query($sql_insert);
do {
    $result = $stmt->fetch(PDO::FETCH_ASSOC);
    echo 'columnCount = ' . var_export($stmt->columnCount(), true) . PHP_EOL;
    echo 'result = ' . var_export($result, true) . PHP_EOL;
} while ($stmt->nextRowset());
```

Временная таблица — особенность работы OUTPUT INSERTED для таблиц с триггерами.

Результаты rowset'ов

SQLSRV

```
columnCount = 1
result = array (
  'id' => '4',
)
```

DBLIB

```
columnCount = 0
result = false

columnCount = 0
result = false

columnCount = 1
result = array (
  'id' => 5,
)
```

нет rowset только от DECLARE

Issue, связанные с поведением:
Column rowsets to be skipped automatically | Empty result set reported for TDS INFO messages

А вы знали, что...

Используя PDO + официальный драйвер от Microsoft, вы не сможете вставить в запрос больше 2100 значений с помощью bindValue.

```
PDOException: SQLSTATE[IMSSP]: Tried to bind parameter number 2101.
SQL Server supports a maximum of 2100 parameters.
```

source/shared/core_sqlsrv.h

```
const int SQL_SERVER_MAX_PARAMS = 2100;
const int SQL_SERVER_MAX_MONEY_SCALE = 4;

// increase the maximum message length to accommodate for the long error returned for
operand type clash
```

https://github.com/microsoft/msphpsql/issues/410

Именно это часто толкает к использованию DBLIB



– Ну как же туда не ходить? Они же ждут!

© Котёнок по имени Гав

Еще мелочей?

Что может пойти не так при обычном SELECT?

PHP 7.4/8.0 + PDO (MySQL)

```
// SELECT * FROM simple_table WHERE id=1
var_dump($result);

/*
array(4) {
  'int_col'     => string(4) "-123"
  'bigint_col'  => string(10) "8817806877"
  'float_col'   => string(11) "-12345.6789"
  'numeric_col' => string(6) "-33.22"
}
*/
```

Обновляем версию PHP

PHP 8.1 + PDO (MySQL)

```
// SELECT * FROM simple_table WHERE id=1
var_dump($result);

/*
array(4) {
  ["int_col"]     => int(-123)
  ["bigint_col"]  => int(8817806877)
  ["float_col"]   => float(-12345.6789)
  ["numeric_col"] => string(6) "-33.22"
}
*/
```

А я хочу как раньше

PHP 8.1 + PDO (MySQL)

```
$pdo = new PDO('mysql:......;charset=utf8', 'username', 'password');
// для версии 8.1
$pdo->setAttribute(PDO::ATTR_STRINGIFY_FETCHES, 1);
// в драйвере до версии 8.0 флаг включен. С 8.1 флаг не влияет.
$pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, 1);
/*
array(4) {
  'int_col'     => string(4) "-123"
  'bigint_col'  => string(10) "8817806877"
  'float_col'   => string(11) "-12345.6789"
  'numeric_col' => string(6) "-33.22"
}
*/
```

Удовлетворим любопытство?

Было

```
#if SIZEOF_ZEND_LONG==4
    if ((L64(2147483647) < (int64_t) lval) || (L64(-2147483648) > (int64_t) lval)) {
        DBG_INF("stringify");
        tmp_len = sprintf((char *)&tmp, "%" PRIi64, lval);
    } else
#endif /* SIZEOF */
    {
        ZVAL_LONG(zv, (zend_long) lval); /* the cast is safe, we are in the range */
    }
}

if (tmp_len) { // ВОТ ЭТО МЕСТО
    ZVAL_STRINGL(zv, tmp, tmp_len);
}
```

Удовлетворим любопытство?

Стало

```
#if SIZEOF_ZEND_LONG==4
    if ((L64(2147483647) < (int64_t) lval) || (L64(-2147483648) > (int64_t) lval)) {
        DBG_INF("stringify");
        ZVAL_STR(zv, zend_i64_to_str(lval));
    } else
#endif /* SIZEOF */
    {
        ZVAL_LONG(zv, (zend_long) lval); /* the cast is safe, we are in the range */
    }
}
```

Удовлетворим любопытство?

```
            tmp_len = sprintf((char *)&tmp, "%" PRIi64, lval);
        } else
#endif /* SIZEOF */
        {
            ZVAL_LONG(zv, (zend_long) lval); /* the cast is safe, w
        }
    }

    if (tmp_len) {
        ZVAL_STRINGL(zv, tmp, tmp_len);
    }
    (*row)+= byte_count;
```

```
            ZVAL_STR(zv, zend_i64_to_str(lval));
        } else
#endif /* SIZEOF */
        {
            ZVAL_LONG(zv, (zend_long) lval); /* the cas
        }
    }

    (*row)+= byte_count;
    DBG_VOID_RETURN;
```

☑ Changes only

Version	Date	Author	Commit Message
65a5c18	13.04.2021, 16:43	Nikita Popov	Add functions to convert i64/u64 to string

Слабая динамическая типизация

declare(strict_types=1);

```
class TypesCheck {
    private int $n = 1;

    public function check(PDO $pdo)
    {
        $stmt = $pdo->query('select int_col from pdo_types');
        $stmt->bindColumn(1, $this->n, PDO::PARAM_INT);
        $this->anyWork($this->n); // int(1)
        $stmt->fetch(PDO::FETCH_BOUND);
        var_dump($this->n); // string(3) "101"
        $this->anyWork($this->n);
        // PHP Fatal error:  Uncaught TypeError: TypesCheck::anyWork()...
    }

    private function anyWork(int $value) { /* any actions */ }
}
```

PDO::ATTR_EMULATE_PREPARES = 1

Транзакции в PDO

- Следите за **PDO::ATTR_ERRMODE** (PDO::ERRMODE_SILENT, etc.)

Транзакции в PDO

- Следите за **PDO::ATTR_ERRMODE** (PDO::ERRMODE_SILENT, etc.)
- DDL в транзакции поддерживают не все DBMS — будет автокоммит (MySQL, Oracle, MS SQL)

Транзакции в PDO

- Следите за **PDO::ATTR_ERRMODE** (PDO::ERRMODE_SILENT, etc.)
- DDL в транзакции поддерживают не все DBMS – будет автокоммит (MySQL, Oracle, MS SQL)
- При старте PDO только проверит, что драйвер их поддерживает и нет активной транзакции

Транзакции в PDO

- Следите за **PDO::ATTR_ERRMODE** (PDO::ERRMODE_SILENT, etc.)
- DDL в транзакции поддерживают не все DBMS – будет автокоммит (MySQL, Oracle, MS SQL)
- При старте PDO только проверит, что драйвер их поддерживает и нет активной транзакции
- Вложенные транзакции не поддерживаются*

*Свои методы для проверки активности транзакции *_handle_in_transaction имеют только PostgreSQL и MySQL (PHP 8.0), и они узнают о транзакции, даже если вы стартовали её через SQL.

Вложенные транзакции

```
if (pdo_is_in_transaction(dbh)) {
  zend_throw_exception_ex(php_pdo_get_exception(), 0,
        "There is already an active transaction");
  RETURN_THROWS();
}
```

```
static bool pdo_is_in_transaction(pdo_dbh_t *dbh) {
        if (dbh->methods->in_transaction) {
                return dbh->methods->in_transaction(dbh);
        }
        return dbh->in_txn; // boolean флаг
}
```

А вот метод из условия: dbh->methods->in_transaction

```
static bool pdo_mysql_in_transaction(pdo_dbh_t *dbh)
{
  pdo_mysql_db_handle *H = (pdo_mysql_db_handle *)dbh->driver_data;
  PDO_DBG_ENTER("pdo_mysql_in_transaction");
  PDO_DBG_RETURN((pdo_mysql_get_server_status(H->server) & SERVER_STATUS_IN_TRANS) != 0);
}
```

Транзакции и MyISAM

```
$pdo1->beginTransaction(); // true
$insertResult1 = $pdo1->prepare($sql_insert)->execute([':n' => 111]);
$pdo1->rollBack(); // true

$st = $pdo1->prepare('SELECT id,int_col FROM type WHERE int_col=:n');
$st->execute(['n' => 111]);
var_dump($st->fetch(PDO::FETCH_ASSOC));
```

```
array(2) {
  'id' => string(1) "3"
  'int_col' => string(3) "111"
}
```

Документация обещает исключение в таких случаях, но мы его не получим, т.к. драйвер ничего не знает про MyISAM-таблицу, ведь БД у нас INNODB.

Транзакции в MSSQL

С использованием DBLIB (FreeTDS driver for Microsoft SQL Server and Sybase databases)

```
$pdo = new PDO('dsn_string', 'username', 'password');
$pdo->beginTransaction();
$pdo->commit();
$pdo->rollBack();
```

С php5-sybase (old driver for MS SQL Server)

```
$pdo = new PDO('dsn_string', 'username', 'password');
$pdo->exec('BEGIN TRANSACTION');
$pdo->exec('COMMIT TRANSACTION');
$pdo->exec('ROLLBACK TRANSACTION');
```

Лучше, конечно же, использовать PDO_SQLSRV, но посмотрите, какие чудесные «костыли».

А еще так можно реализовать вложенные транзакции...

История про PDO::quote и ¿'

The payload we're going to use for this injection starts with the byte sequence `0xbf27`. In gbk, that's an invalid multibyte character; in latin1, it's the string `¿'`. Note that in latin1 and gbk, 0x27 on its own is a literal `'` character.

ext/pdo_mysql/mysql_driver.c

```
if (use_national_character_set) {
    *quotedlen = mysql_real_escape_string_quote(H->server, *quoted + 2, unquoted,
        unquotedlen, '\'');
    (*quoted)[0] = 'N';
    (*quoted)[1] = '\'';

    ++*quotedlen; /* N prefix */
} else {
    *quotedlen = mysql_real_escape_string_quote(H->server, *quoted + 1, unquoted,
        unquotedlen, '\'');
    (*quoted)[0] = '\'';
}

(*quoted)[++*quotedlen] = '\'';
(*quoted)[++*quotedlen] = '\0';
```

0x-теричная система счисления

Можно ли вставить строку прямо в запрос и не бояться уязвимостей?

Ответ: да, если у вас MS SQL

```
// Данные заэкранируем хексом - фича msskl
$string = '0x' . bin2hex($value);
$sql = "INSERT INTO [table]([varchar_col]) VALUES($string)";
```

0x-теричная система счисления

Можно ли вставить строку прямо в запрос и не бояться уязвимостей?

Ответ: да, если у вас MS SQL

```
// Данные заэкранируем хексом - фича мсскл
$string = '0x' . bin2hex($value);
$sql = "INSERT INTO [table]([varchar_col]) VALUES($string)";
```

Ну и теперь сохраняем строковые данные в BLOB поле MS SQL

```
if (is_string($value)) {
    return new Expression('CONVERT(VARBINARY(MAX), '.
            ('0x'. bin2hex($value)) .')');
}
```

Спасибо Андрею Рычкову за науку

Ну его нафиг...



Так ведь и выгореть можно.

Вместо вывода

Все преимущества ORM, мокрые от слёз разработчиков

- Код может быть абстрактным
- Нет нужды писать «сырые» запросы
- Порог вхождения гораздо ниже
- Код не слишком привязан к движку БД

Теперь, если вам понадобится своя ORM — вы будете готовы

Вместо вывода

Все преимущества ORM, мокрые от слёз разработчиков

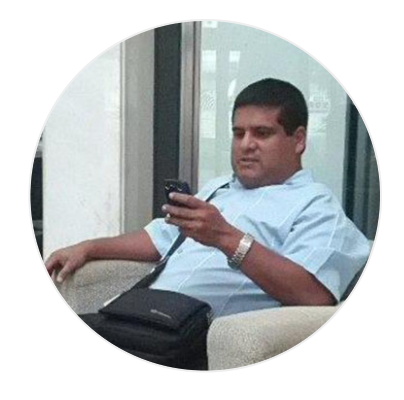
- Код может быть абстрактным
- Нет нужды писать «сырые» запросы
- Порог вхождения гораздо ниже
- Код не слишком привязан к движку БД

Теперь, если вам понадобится своя ORM — вы будете готовы**, наверное.**

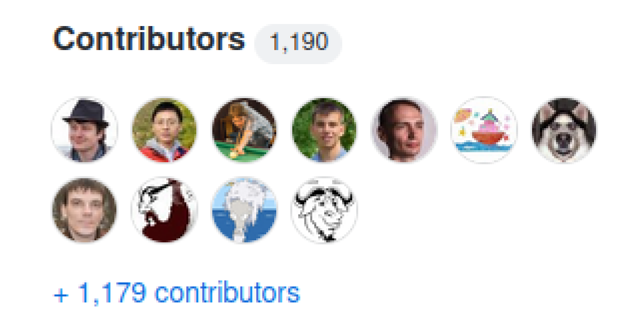
Благодарности



Александр Макаров



Wilmer Arambula



И всем, кто вносит свой вклад в OpenSource

Литература и материалы

- https://phpdelusions.net/pdo
- https://phpdelusions.net/pdo/fetch_modes
- https://www.php.net/manual/ru/refs.database.php
- https://docs.microsoft.com/ru-RU/sql/t-sql/functions/scope-identity-transact-sql?view=sql-server-2016
- https://habr.com/ru/post/141127/
- https://www.sqlitetutorial.net/sqlite-describe-table/
- https://habr.com/ru/post/137664
- https://habr.com/ru/company/vk/blog/234125/
- https://stackoverflow.com/questions/134099/are-pdo-prepared-statements-sufficient-to-prevent-sql-injection/
- Иконки от Freepik - Flaticon. https://www.flaticon.com/ru/free-icons/
- https://github.com/yiisoft/yii2/issues/18171
- https://stackoverflow.com/questions/13747481/pdo-lastinsertid-not-working-for-ms-sql
- https://docs.microsoft.com/ru-RU/sql/t-sql/functions/scope-identity-transact-sql?view=sql-server-2016
- https://bugs.php.net/bug.php?id=69592
- https://github.com/FreeTDS/freetds/issues/156
- https://github.com/microsoft/msphpsql/issues/410
- https://mneti.ru
- и, конечно же, официальная документация к *sql drivers

Спасибо, что выслушали.

Голосуйте за мой доклад

→ → →



Валерий Горбачев
https://slides.darkdef.com/pdo/

PHP Russia 2022